Finanziato dall'Unione europea
NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani
PIANO NAZIONALE DI RIPRESA E RESILIENZA
ICSC
Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

The Spoke 2 of the ICSC National Centre, with a focus on deep learning applications in astroparticle physics and satellite imagery

G.A. Anastasi
Università di Catania & INFN-Catania

13th CRIS-MAC, Trapani, 17-21 June 2024

The National Centre for HPC, Big Data and Quantum Computing

Managed by the [ICSC Foundation](), one of the five National Centres founded under the Italian PNRR.

Goal : a long-term, national, distributed infrastructure for cutting-edge research and innovation in high-performance and high-throughput computing.

A total investment of almost 320 million Euros
Sept. 2022 – Aug. 2025

Over 50 founding members, to foster synergy between scientific and industrial sectors.

ENTI E UNIVERSITÀ

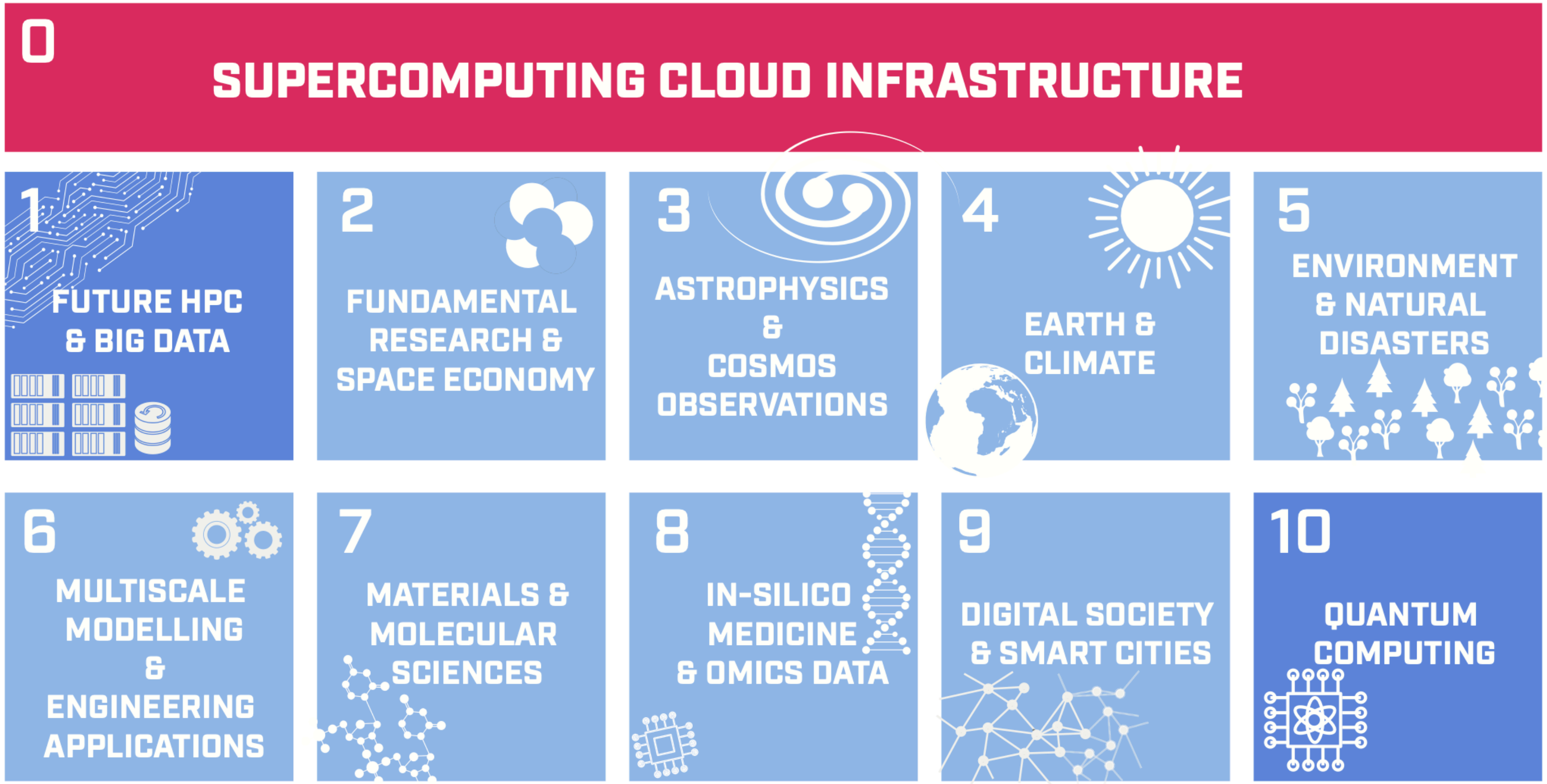
CINECA, Consiglio Nazionale delle Ricerche, CMCC Centro Euro-Mediterraneo sui Cambiamenti Climatici, CRS4, ENEA, Fondazione Bruno Kessler, Consortium GARR, IIT Istituto Italiano di Tecnologia, INAF Istituto Nazionale di Astrofisica, INFN Istituto Nazionale di Fisica Nucleare, INGV, OGS, Politecnico di Bari, Politecnico Milano 1863, Politecnico di Torino, SISSA, Scuola Normale Superiore, Sapienza Università di Roma, Università degli Studi dell'Aquila, Università degli Studi di Bari Aldo Moro, Alma Mater Studiorum Università di Bologna, Università della Calabria, Università di Catania, Università degli Studi di Napoli Federico II, Università degli Studi di Ferrara, Università degli Studi Firenze, Università degli Studi di Milano Bicocca, Unimore, Università degli Studi di Padova, Università di Parma, Università di Pavia, Università di Pisa, Tor Vergata, Università del Salento, Università di Torino, Università di Trento, Università degli Studi di Trieste

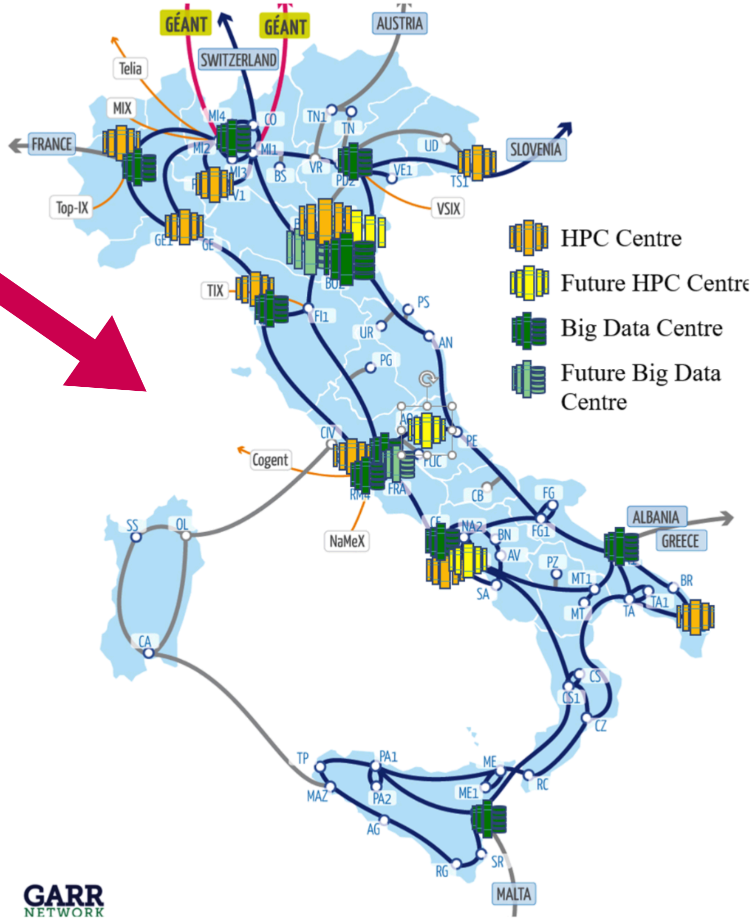
AZIENDE E ISTITUTI PRIVATI

Enel, Engineering, Eni, Ferrovie dello Stato Italiane, Fincantieri, fondazione innovazione urbana, autostrade per l'italia, Humanitas Research Hospital, iFAB, Intesa Sanpaolo, Leonardo, Sogei, Thales Alenia Space, Terna Driving Energy, UnipolSai Assicurazioni

The Spoke and hub model

One cross spoke, Spoke 0 ("Supercomputing Cloud Infrastructure"), and 10 thematic spokes.





Spoke 2 - Fundamental Research & Space Economy

Activities coordinated by INFN and organized around **6 Work Packages** :

WP1: algorithms and tools for theor. physics	WP2: applications for exp. high-energy physics	WP3: applications for exp. astroparticle and G.W.	*"scientific" WPs*
WP4: boost computational performances & porting to GPU, FPGA, etc.	WP5: support for data management & distributed data-lake infrastructure	WP6: Cross Domain initiatives & Space Economy	*"technical" WPs*

Many objectives, such as:

- design, develop, and test solutions suited to both current and next-generation experiments;
- provide new tools with applications beyond science, as in many historical examples (*WWW*, *Grid*, ...);
- proliferate HPC methodologies across Italian academic and industrial sectors.

A WP3 use-case :
DAIDREAM

The use-case DAIDREAM within WP3

***DA**ta-driven **ID**entification of **R**are **E**vents in **A**stroparticle physics through **M**achine learning techniques*

Employ **self-supervised or weakly supervised deep-learning** to fully exploit experimental data (examples in anomaly detection for HEP and rejection of noise transients in GW)

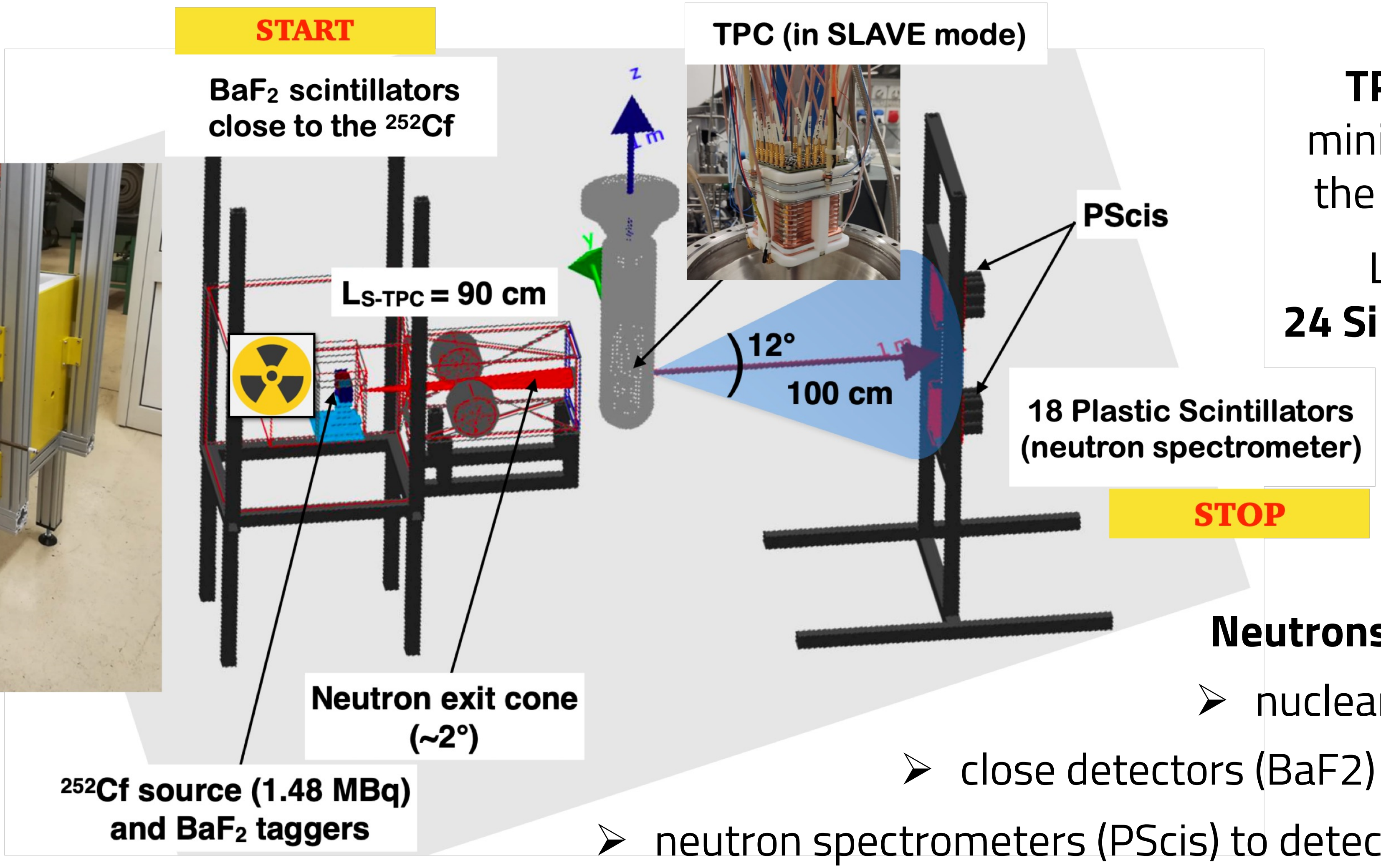
Application in at least 2 distinct (yet contiguous) experimental settings:

- searches for WIMPs with mass up to ~10 TeV in dual-phase Liquid Argon TPCs
- search for rare or anomalous air-shower footprints in ground-based observatories

Project developed on the **INFN-Cloud** infrastructure, in particular using isolated Virtual Machines running JupyterHub with Notebooks persistence.

The Recoil Directionality (ReD) experiment

Eur. Phys. J. C 81 (2021) 11, 1014



TPC (5 x 5 x 6) cm^3
miniaturized version of the DarkSide-20k TPC

Light readout by
24 SiPMs in 5x5 cm^2 tiles

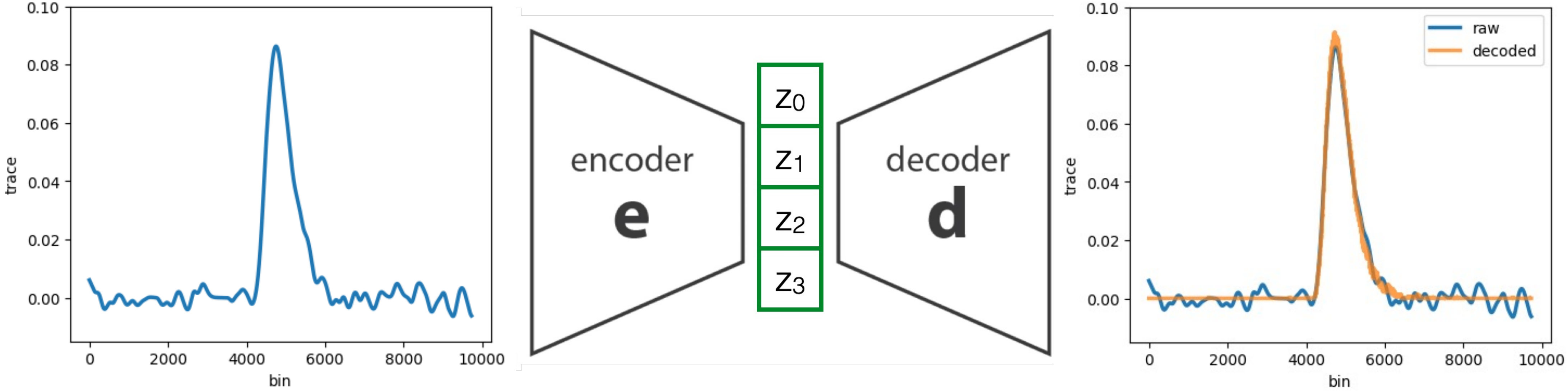
Neutrons from a ^{252}Cf source

- nuclear recoils of a few keV
- close detectors (BaF2) to tag fission events
- neutron spectrometers (PScis) to detect scattered neutrons

Convolutional AutoEncoders and their application

Self-supervised neural network architecture where data are compressed into a low dimensionality *latent space*, then reconstructed minimizing differences between original and output.

Implicitly highlighting features of a dataset, while disregarding noise and redundancies

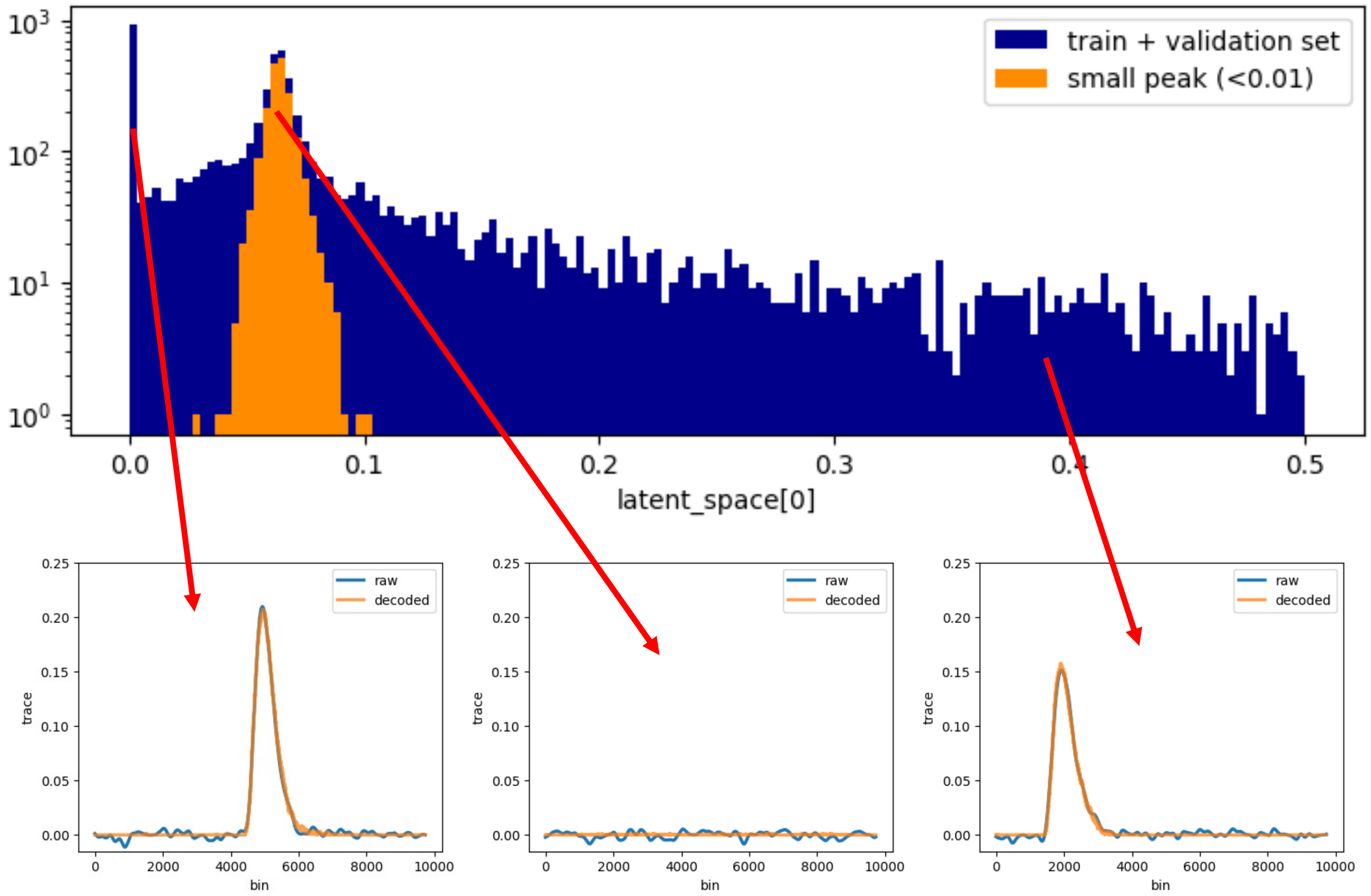


- **input**: time series (~10,000 bins) resembling waveforms measured by the ReD TPC
- **architecture**: 3 Conv1D + avg. pooling layers, followed by 1 flattened dense layer (*details in backup*)
- **4-dimensional latent space** (i.e. each trace is compacted into only 4 values, named z_i)

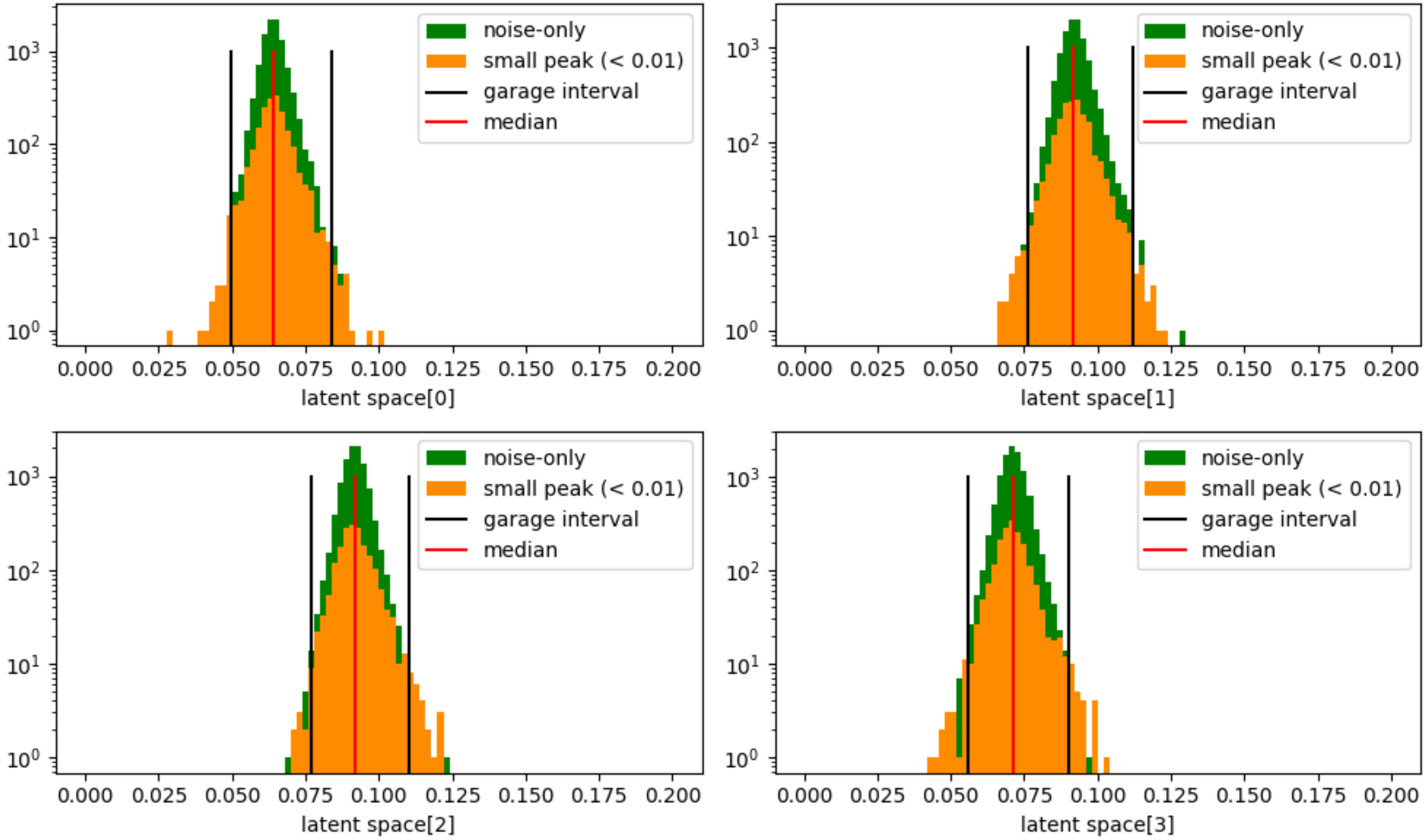
Application to a synthetic dataset

Synthetic waveforms: single-peaked log-normal shaped signal on top of <u>non-gaussian</u> noise.
Generated signals have amplitude (i.e. peak value) distributed in [~0, 1].

Characterizing result:
waveforms with **negligible signals** encoded into a limited region of the latent space (nicknamed as "*garage*") where the 4 $\boldsymbol{z}_i$ <u>simultaneously</u> assume specific values.



Application to a synthetic dataset - results



Garage defined as the combination of the 3σ-intervals around median calculated for each z_i distribution using "noise-only" waveforms

False positives fraction < 1%

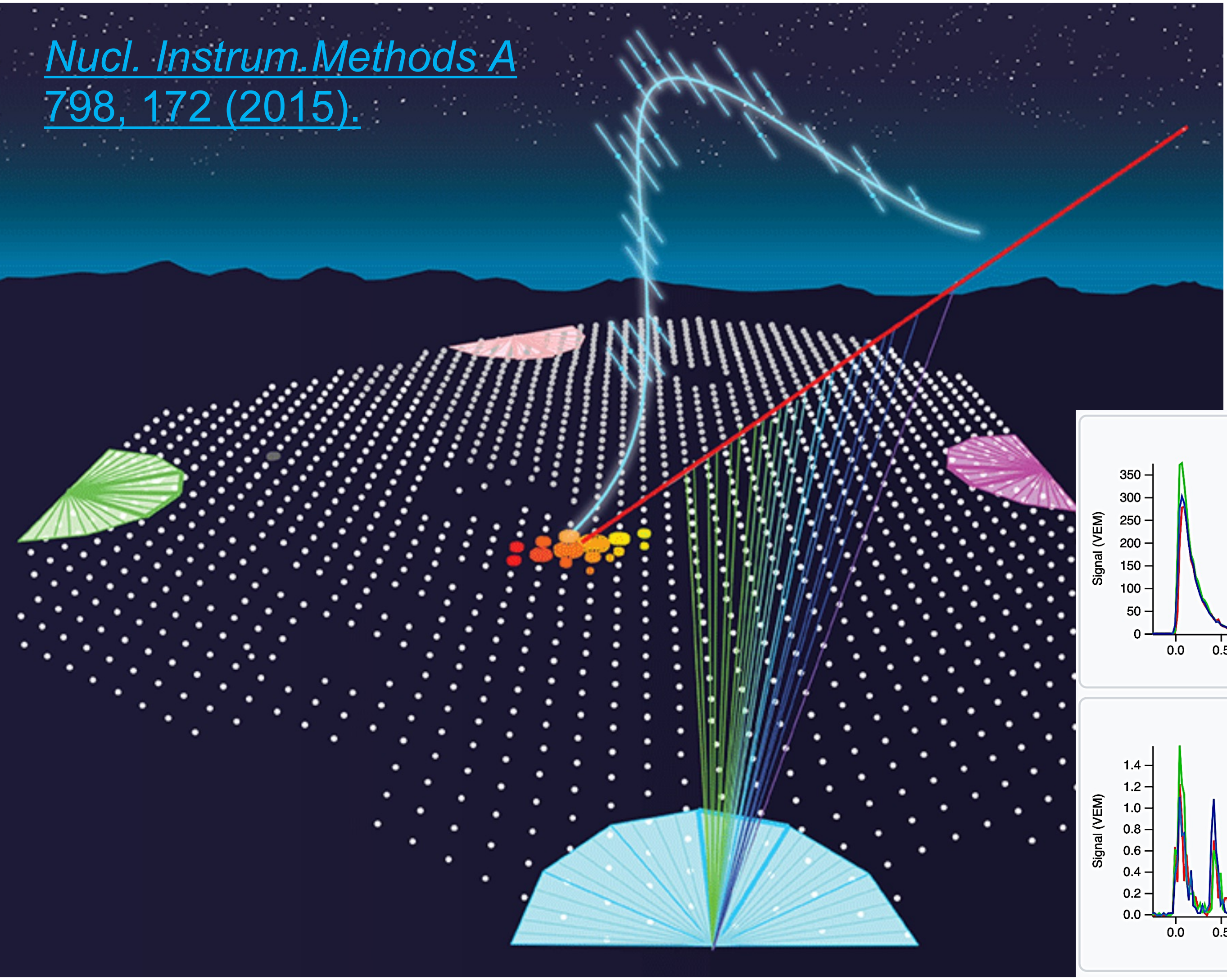
Labelling of events :

- if the 4 z_i fall simultaneously in the "*garage*", tag as noise-only;
- if not, tag as signal.

True positives fraction > 99% down to signal amplitudes ~ 0.015

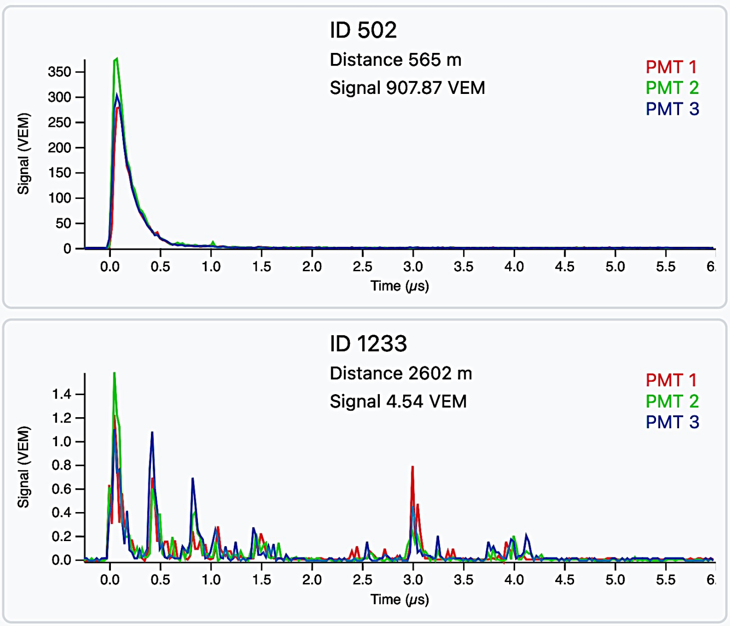
Thanks to **N. Pino, S. Puglia, S. Albergo** (UniCT)
for their contributions in developing this methodology.

The Pierre Auger Observatory



Largest experiment for the detection of ultra-high energy cosmic rays.

Deep-learning applications focus on exploiting the extensive air-shower footprints at ground level, measured by the Surface Detector (SD), **to infer mass-sensitive information**.



See for example:

JINST 16 (2021) P07016

JINST 16 (2021) P07019

arXiv:2406.06319

Typical signals acquired by the water-Cherenkov tanks of the SD array

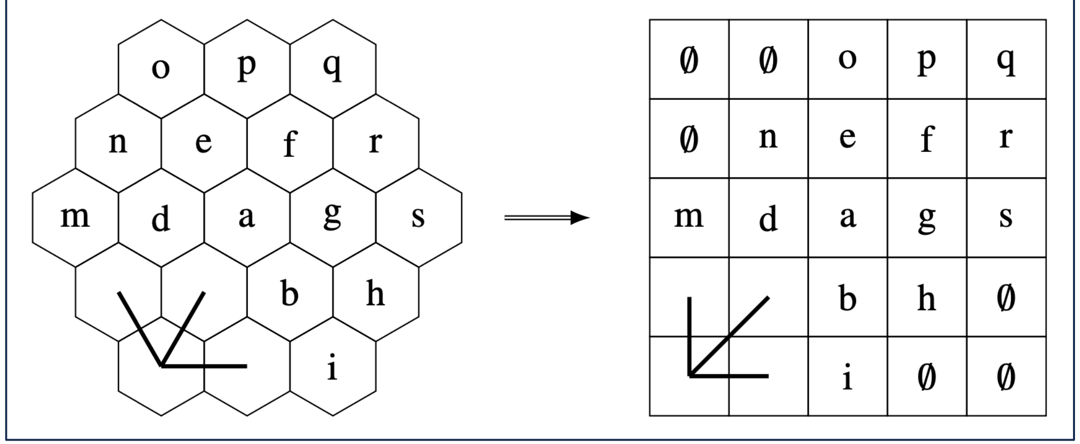
Applications to the Pierre Auger Observatory data

Goal: identification of unusual events, possibly induced by non-hadronic primary particles.

Strategy: process the temporal and spatial structure of the footprint at ground level with deep convolutional neural networks, separating the space and time measurements.

Fundamental step in the data pre-processing:
transformation (i.e. re-indexing) of station locations from the SD triangular grid into a Cartesian grid.
Astropart. Phys. 97, 46 (2018)

PoS(ICRC2021)239



Proposal : preliminary analysis using **Auger Open Data**
> 25000 events with primary energy above ~2.5 x 10^{18} eV , corresponding to 10% of the dataset used in the Auger physics analyses presented at the *International Cosmic Ray Conference* in 2019.

A WP6 flagship use-case:
AI algorithms for (satellite) imaging reconstruction

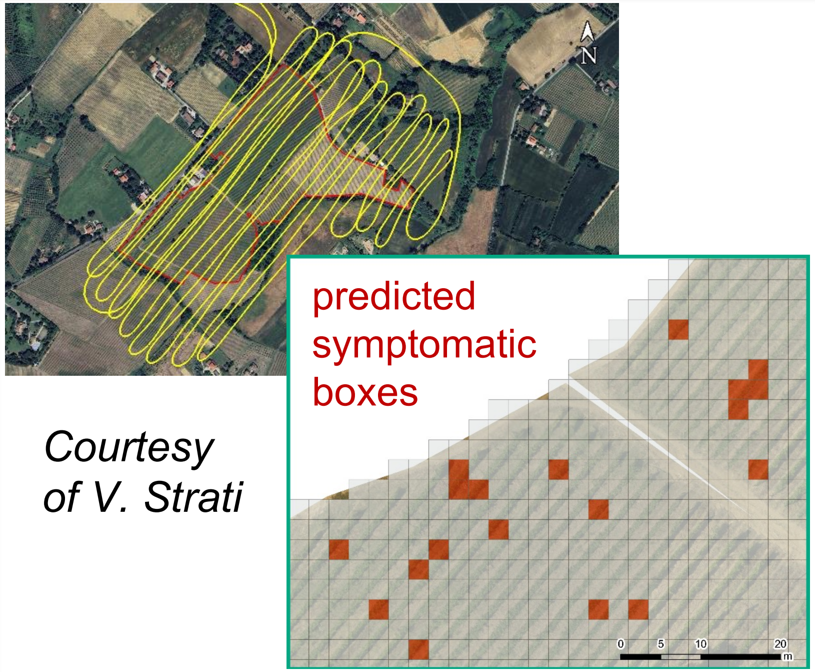
Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
ICSC Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

Space Economy within the WP6

Participating institutions: UniFE, UniCT, INFN-Catania

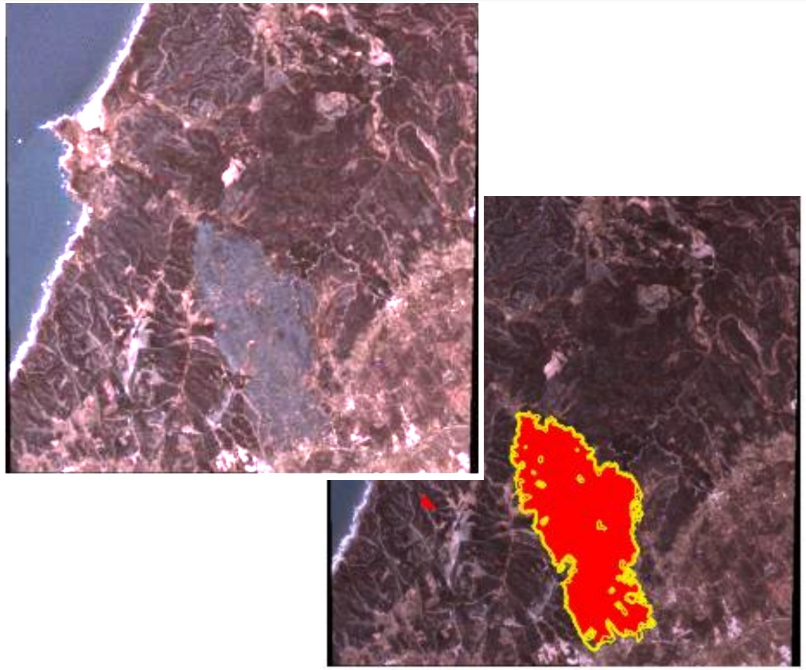
INFN CATANIA Istituto Nazionale di Fisica Nucleare Sezione di Catania

Deterministic Learning algorithms for object identification of photovoltaic panels in aerial images.
Technologies 2023, 11, 174



Courtesy of V. Strati

Disease detection in vineyards using high-resolution images collected by Unmanned Aerial Vehicles (UAVs).
V. Strati, EGU24-10773 (2024)



Analysis of satellite imagery using deep-learning for disease detection in vineyards and **segmentation of wildfire-affected areas**.

Sentinel-2 mission and the *Copernicus Emergency Management Service*

The **Copernicus Sentinel-2 mission**:
twin satellites for high-resolution, high revisit frequency, multi-spectral imaging.

Copernicus Emergency Management Service (**CEMS**) :
one of six services within the Earth Observation component of the European Union's space programme.

Head of development :
G. Piparo (INFN-CT)

Downloading and (pre-)processing Sentinel-2 data

Within the project, a ***python*** library has been developed, currently including of 4 modules:

Sentinel Download	Download of satellite imagery using the Sentinel-Hub API. Currently implemented for Sentinel2-L2A products only.
Sentinel DataManipulator	Produce maps for single spectral band and vegetation indexes (currently 19 implemented) in TIFF format and as ***numpy*** arrays. Also combining downloaded data with wildfires information from CEMS (if available).
Sentinel Visualiser	Printing the processed maps in standard formats (PDF, PNG, etc.)
Sentinel DataHandling	Pre-processing of data for training deep-learning applications. Currently includes : dataset normalization, discrete mirroring/rotations & image splitting for data augmentation, storage in csv or numpy-native formats.

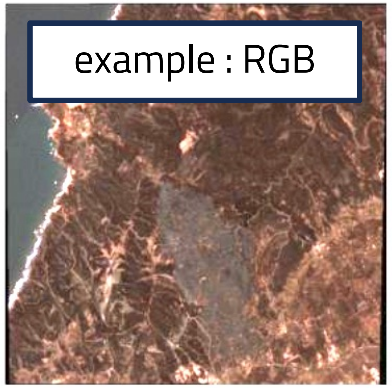
To be made publicly available as open-source library by the end of the project.

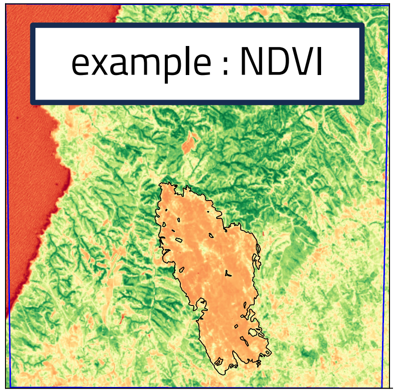
Segmentation of wildfire-affected areas

Thanks to G. Piparo for the deployment of the architecture

Goal: delimitation of burnt areas using supervised Convolutional Neural Networks (*a review of deep learning applications in this field*) trained on wildfire activation maps from CEMS.

Preliminary dataset : 23 areas, 512x512 pixels, 3 different times (30 to 10 days before / during / 10 to 30 days after the event).



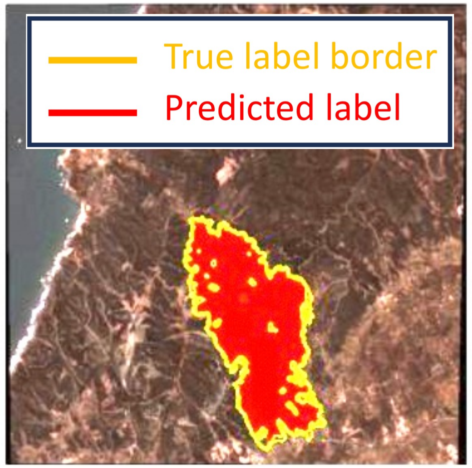


Features selection: 10 spectral bands + 9 vegetation indexes.

Architecture : UNet-like with Long Short Term Memory (LSTM) layers + custom losses (Dice + Jaccard + cross-entropy).

Satisfying preliminary results, but a lot of room for improvement:

larger dataset, better selection of events, download at higher resolution, better cloud management, data augmentation tools, …



What's next ?

Activities currently underway

Research plan in 4 phases :

(1) landscape recognition (1 year – concluded) **(2) realization (currently underway)**

(3) validation of developed tools & algorithms (4) wrap-up & dissemination

WP3 – DAIDREAM

- Apply method to ReD measurements (very promising preliminary results)
- Combine tools for processing of multi-layered maps with autoencoder applied on waveforms

WP6 - AI algorithms for (satellite) imaging reconstruction

Segmentation of wildfire-affected areas:

- ❑ Download full dataset (up to 178 events) at higher granularity (2048 x 2048 pixels)
- ❑ Perform data-augmentation during training
- ❑ Study different architectures

Conclusions

The National Centre for HPC, Big Data and Quantum Computing is a great opportunity for research in Italy, going well beyond securing cutting-edge computing resources and technologies:

- form a globally appealing ecosystem, based on public-private partnerships;
- address both current and emerging scientific and societal challenges;
- training of a new generation of computing-savvy researchers and Ph.D. graduates.

Supercomputing shaping the future

Thanks to the **ReD & Darkside Collaborations** and to the **Pierre Auger Collaboration** for providing precious information about the apparatuses and the measurements formats.

Backup

Convolutional AutoEncoder (CAE) for the analysis of waveforms

Training : Keras with **Tensorflow** as backend

Optimizer : ***ADAM*** - initial Learning Rate 0.001

Loss function : sum of square differences btw input and output, equivalent to **MSE x number of time-bins** (= 9728)

Activation function : ***ReLu*** (after Conv. & Dense layers)

Callback : ***ReduceLROnPlateau***
monitor='val_loss', factor=0.5,
patience=5, cooldown=5, min_delta=1e-4,
restore_best_weights=True

batch size = 100 (fixed)

Dataset : 10,000 synthetic waveforms
- 7500 (6000+1500) training + validation
- 2500 testing

Encoder

Latent Space
z_i variables

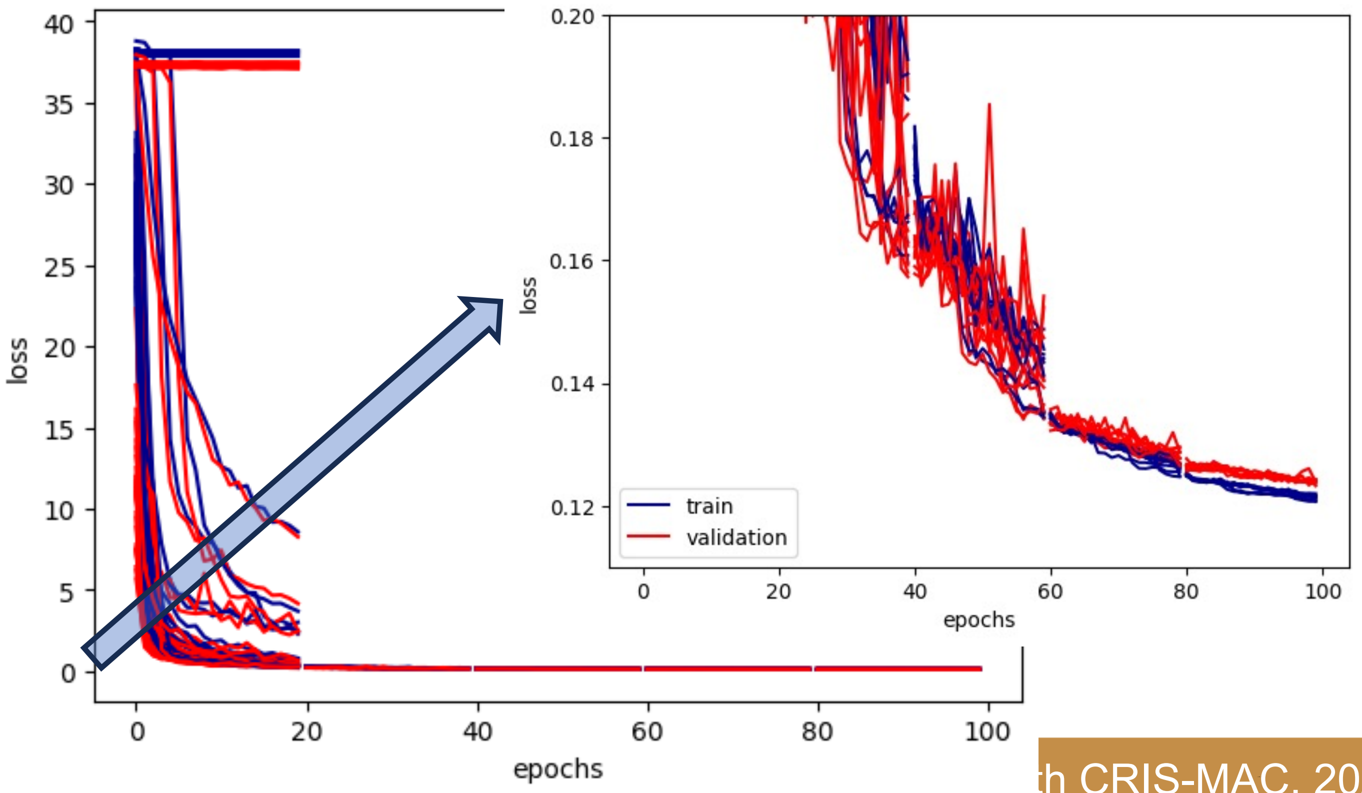
Decoder

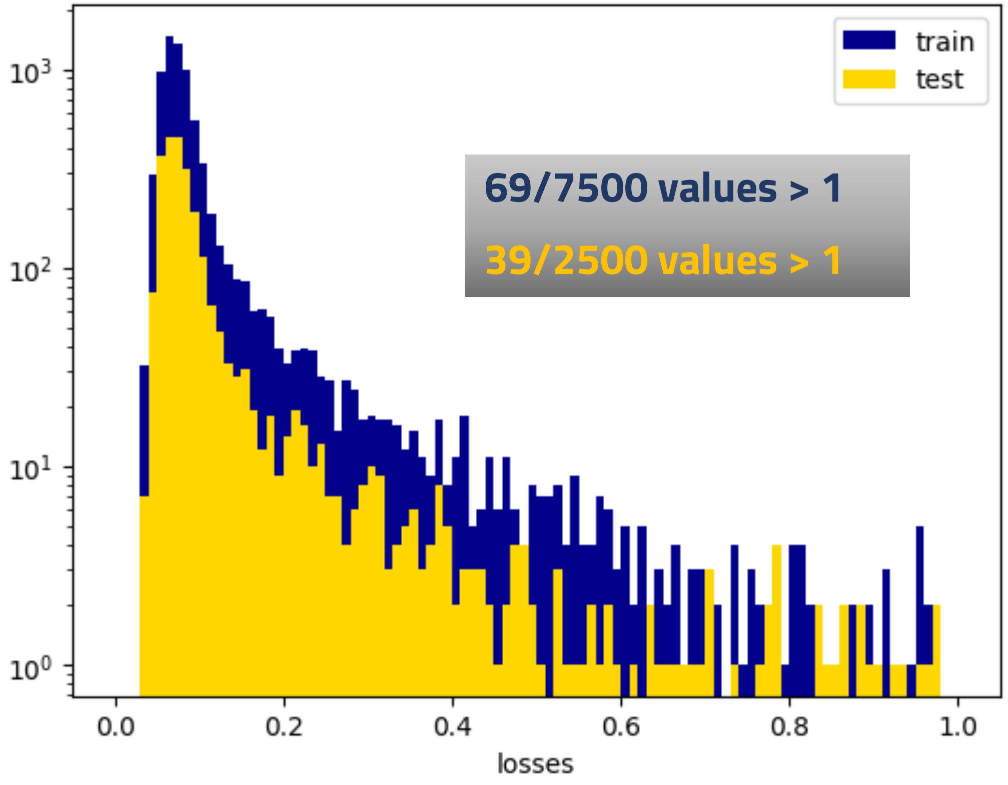
Layer (type)	Output Shape	Param #
conv1 (Conv1D)	(None, 4864, 2)	66
average_pooling1d_18 (AveragePooling1D)	(None, 2432, 2)	0
conv2 (Conv1D)	(None, 1216, 4)	260
average_pooling1d_19 (AveragePooling1D)	(None, 608, 4)	0
conv3 (Conv1D)	(None, 304, 8)	1032
average_pooling1d_20 (AveragePooling1D)	(None, 152, 8)	0
flatten (Flatten)	(None, 1216)	0
dense_encoded (Dense)	(None, 4)	4868
dense_decoded (Dense)	(None, 1216)	6080
reshape (Reshape)	(None, 152, 8)	0
up_sampling1d_18 (UpSampling1D)	(None, 304, 8)	0
deconv3 (Conv1DTranspose)	(None, 608, 4)	1028
up_sampling1d_19 (UpSampling1D)	(None, 1216, 4)	0
deconv2 (Conv1DTranspose)	(None, 2432, 2)	258
up_sampling1d_20 (UpSampling1D)	(None, 4864, 2)	0
deconv1 (Conv1DTranspose)	(None, 9728, 1)	65

Total params: 13657 (53.35 KB)
Trainable params: 13657 (53.35 KB)
Non-trainable params: 0 (0.00 Byte)

CAE training scheme

1. train 50 models with different random initialization for 20 epochs, then select model with lowest validation loss
2. train best model from step (1) for 20 epochs restarting the optimizer with a different seed 10 times, then select model with lowest validation loss
3. repeat step (2) four times, for a total of 100 epochs (20 x 4 + the initial 20 epochs from step (1))





Finanziato dall'Unione europea
NextGenerationEU
Ministero dell'Università e della Ricerca

UNet-like CNN with LSTM layers

Optimizer : ***ADAM*** - initial Learning Rate 0.001

Custom loss function (see box below)

Test training performed for 50 epochs, batch_size = 2 with 23 time series (3 images of 512x512 pixels each)
70% in training set, 30% in test set

```
def create_lstm_unet(input_shape, weight_decay=1e-4, callbacks=None):
    inputs = Input(shape=input_shape)

    # Encoder
    c1 = ConvLSTM2D(16, (3, 3), padding='same', kernel_regularizer=l2(weight_decay), return_sequences=True)(inputs)
    #c1 = TimeDistributed(BatchNormalization())(c1)
    c1 = Activation('relu')(c1)
    c1 = TimeDistributed(Dropout(0.2))(c1)
    p1 = TimeDistributed(MaxPooling2D((2, 2)))(c1)

    c2 = ConvLSTM2D(32, (3, 3), padding='same', kernel_regularizer=l2(weight_decay), return_sequences=True)(p1)
    #c2 = TimeDistributed(BatchNormalization())(c2)
    c2 = Activation('relu')(c2)
    c2 = TimeDistributed(Dropout(0.2))(c2)
    p2 = TimeDistributed(MaxPooling2D((2, 2)))(c2)

    # Bottleneck
    bn = ConvLSTM2D(64, (3, 3), padding='same', kernel_regularizer=l2(weight_decay), return_sequences=True)(p2)
    #bn = TimeDistributed(BatchNormalization())(bn)
    bn = Activation('relu')(bn)
    bn = TimeDistributed(Dropout(0.2))(bn)

    # Decoder
    u1 = TimeDistributed(UpSampling2D((2, 2)))(bn)
    u1 = ConvLSTM2D(32, (3, 3), padding='same', kernel_regularizer=l2(weight_decay), return_sequences=True)(u1)
    #u1 = TimeDistributed(BatchNormalization())(u1)
    u1 = Activation('relu')(u1)
    u1 = TimeDistributed(Dropout(0.2))(u1)
    concat1 = Concatenate(axis=-1)([u1, c2])

    u2 = TimeDistributed(UpSampling2D((2, 2)))(concat1)
    u2 = ConvLSTM2D(16, (3, 3), padding='same', kernel_regularizer=l2(weight_decay), return_sequences=True)(u2)
    #u2 = TimeDistributed(BatchNormalization())(u2)
    u2 = Activation('relu')(u2)
    u2 = TimeDistributed(Dropout(0.2))(u2)
    concat2 = Concatenate(axis=-1)([u2, c1])

    # Output layer
    outputs = ConvLSTM2D(1, (3, 3), activation='sigmoid', padding='same', return_sequences=True)(concat2)
    outputs = Lambda(lambda x: x[:, -1, :, :, :])(outputs)  # Last timestep

    model = Model(inputs=inputs, outputs=outputs)
    model.compile(optimizer=Adam(learning_rate=0.001), loss=dice_jaccard_crossentropy_loss, metrics=['accuracy'])

    return model
```

CUSTOM LOSS FUNCTIONS

```
def dice_loss(y_true, y_pred):
    y_true_f = K.cast(y_true, 'float32')
    y_pred_f = y_pred
    numerator = 2 * K.sum(y_true_f * y_pred_f)
    denominator = K.sum(y_true_f + y_pred_f)
    return 1 - (numerator + K.epsilon()) / (denominator + K.epsilon())

def jaccard_loss(y_true, y_pred):
    y_true_f = K.cast(y_true, 'float32')
    y_pred_f = y_pred
    intersection = K.sum(y_true_f * y_pred_f)
    sum_ = K.sum(y_true_f + y_pred_f)
    jac = (intersection + K.epsilon()) / (sum_ - intersection + K.epsilon())
    return 1 - jac


def dice_jaccard_loss(y_true, y_pred):
    return 0.5 * dice_loss(y_true, y_pred) + 0.5 * jaccard_loss(y_true, y_pred)


def dice_jaccard_crossentropy_loss(y_true, y_pred, dice_weight=0.4, jaccard_weight=0.4, crossentropy_weight=0.2):
    dice_l = dice_loss(y_true, y_pred)
    jaccard_l = jaccard_loss(y_true, y_pred)
    crossentropy_l = binary_crossentropy(y_true, y_pred)
    total_loss = (dice_weight * dice_l) + (jaccard_weight * jaccard_l) + (crossentropy_weight * crossentropy_l)
    return total_loss
```